High performance computing in fish school simulation by vortex sheet method

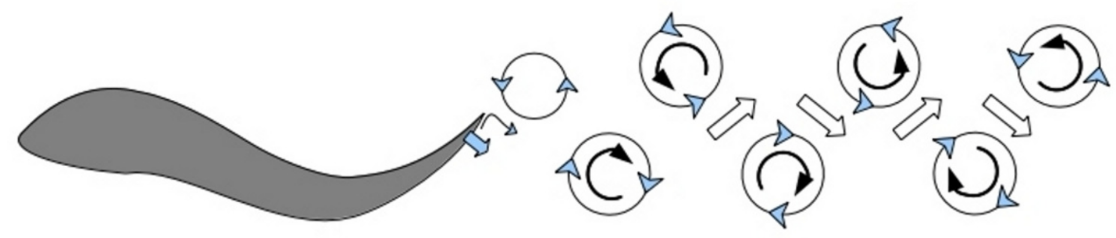
Fang Fang

Applied Math Lab@Courant Inst.

May 19, 2015

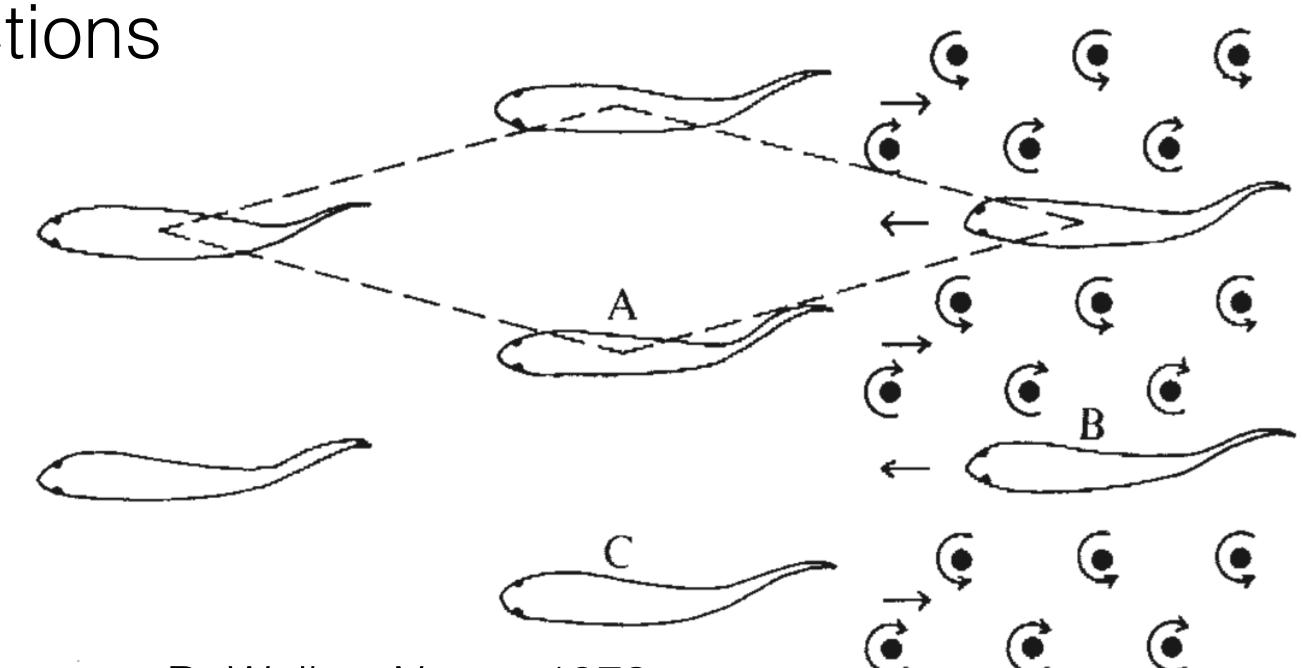
Hydrodynamics of fish schooling — 2D modeling

- Fish propels itself by shedding fluid vortices



J. Wills, *PeerJ* 2003

- School simulation is complicated — fluid-structure interaction, multi-body interactions



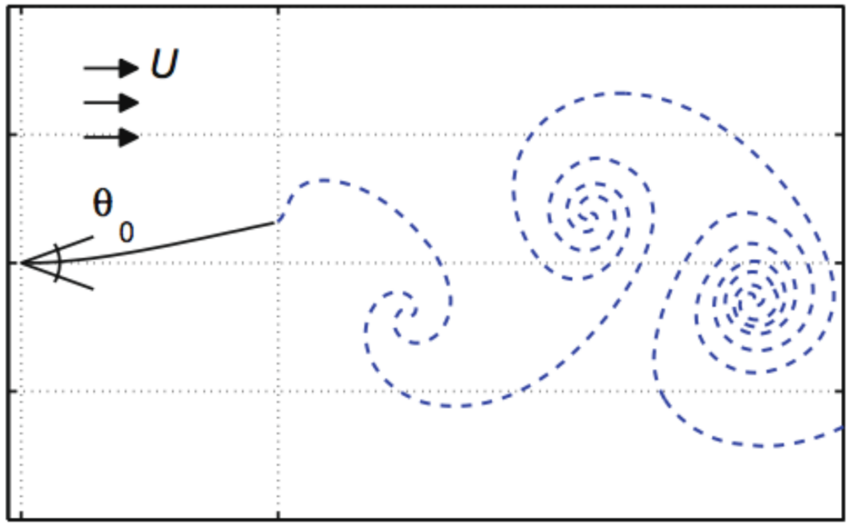
D. Weihs, *Nature* 1973

- School simulation is expensive

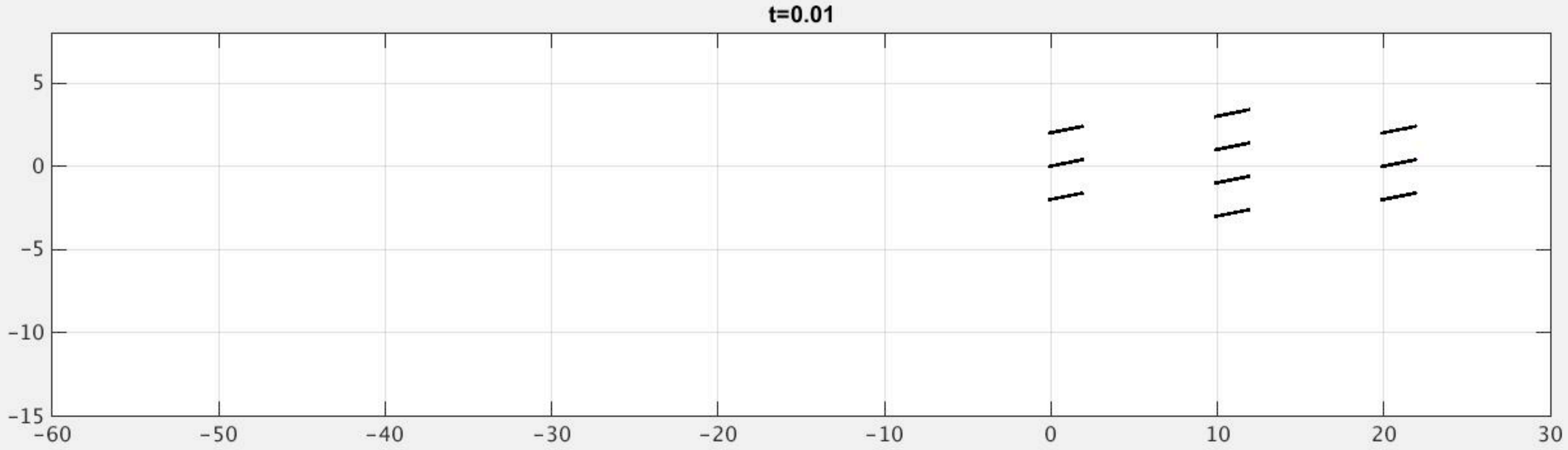
— Number of fish in a school can be large, N~1000 or more

A fish school model using vortex sheet method

- Each fish is modeled as a 1D flapping plate
- Prescribed flapping motion w.r.t. fish head
- Individual fish swims freely under hydrodynamic force and torque
- Fish sheds vorticity (vortex sheet) into fluid from tail, vortex sheet develops in flow



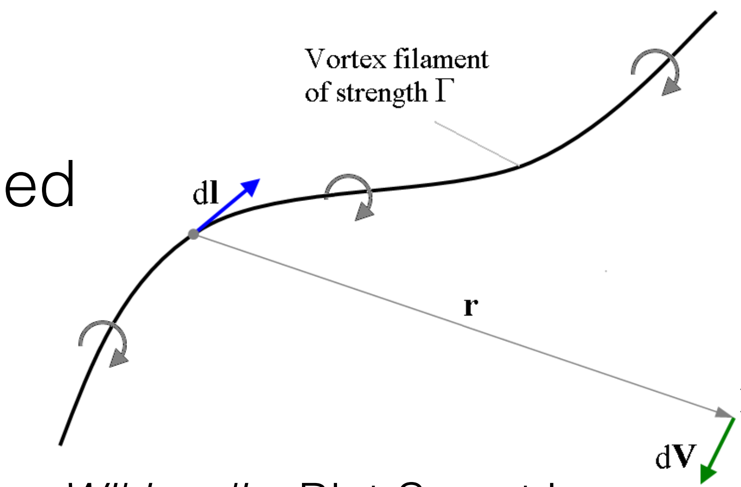
S. Alben, *JCP* 2009



Sketch of simulation algorithm

- **Biot-Savart law:** Evaluate fluid velocity field $\mathbf{u} = (u, v)$ induced by vortex sheets

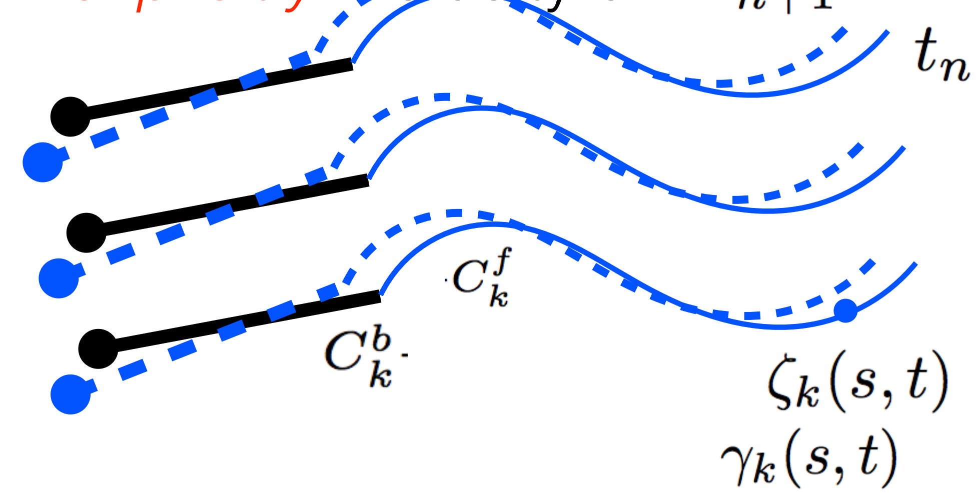
$$\overline{w(z,t)} = u(z,t) - iv(z,t) = \frac{1}{2\pi i} \sum_{k=1,\ldots,N} \int_{C_k^b + C_k^f} \frac{\gamma_k(s,t)}{z - \zeta_k(s,t)} ds. \quad (1)$$



Wikipedia, Biot-Savart Law

- From time step $t_n \to t_{n+1}$:

Step 1. Update positions of free vortex sheets $\zeta_k(s,t)$ *explicitly*. Velocity of sheets w^n are evaluated through (1).



Expensive!

$$\zeta^{n+1} = \zeta^n + \frac{3}{2}\Delta t w^n - \frac{1}{2}\Delta t w^{n-1}, \quad w = u + iv$$

Step 2. Through a nonlinear quasi-Newton solver, *implicitly* solve for fish position $\mathbf{X}_k$, orientation $\mathbf{p}_k$, vorticity distribution, and vortex shedding.

$$\mathbf{u} \cdot \hat{\mathbf{n}} = \dot{\mathbf{x}} \cdot \hat{\mathbf{n}}, \quad Re\left(\overline{(w - \dot{\zeta})}\hat{n}\right) = 0$$

$$m\ddot{\mathbf{X}}_k = F_k, \quad I\ddot{\mathbf{p}}_k = T_k$$

$\Rightarrow$

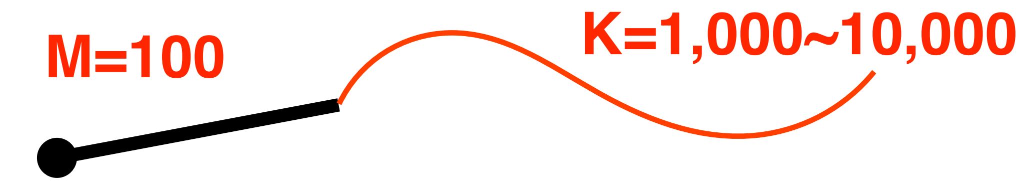
$$Re\left(\overline{(w^{n+1} - \dot{\zeta}^{n+1})}\hat{n}^{n+1}\right) = 0$$

Expensive!

$$\frac{\mathbf{X}^{n+1} - \mathbf{X}^n}{\Delta t} = \frac{1}{2}\left(\dot{\mathbf{X}}^{n+1} + \dot{\mathbf{X}}^n\right),$$

$$m\frac{\dot{\mathbf{X}}^{n+1} - \dot{\mathbf{X}}^n}{\Delta t} = \frac{1}{2}\left(\mathbf{F}^{n+1} + \mathbf{F}^n\right).$$

Parallelism of the *explicit* step



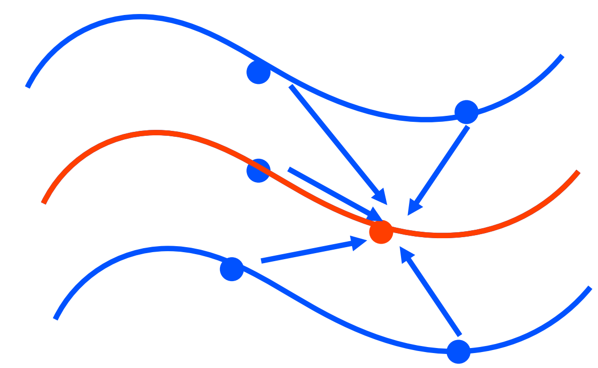
\# of fish: **N=10~1,000**
\# of points on each fish: **M=100**
\# of points on each vertex sheet: **K=1,000~10,000**

→ Total body points: **N*M=1,000~1e5**
Total vortex sheet points: **N*K=1e4~1e7**

Parallel evaluations of vortex sheets velocity (**NK** targets, **NM+NK** sources)

- Induced by sheets — FMM summation over kernel 1/z, **NK** points self-interaction

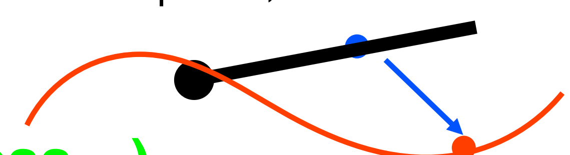
  O(N^2K^2) (direct) ⟶ **O(NK*logNK) (FMM)** ✔



- Correct self sheet interaction with smoothing kernel $\frac{\bar{z}}{|z|^2+\delta^2}$ parallelize **O(K^2)** over **N** fish

  O(NK^2) (serial) ⟶ **O(K^2) (parallel)** ✔ **(tests shown later)**

- Induced by fish bodies — near singular quadrature, singularity subtraction techniques, parallelize **O(MK)** over **N** fish and **N** sheets

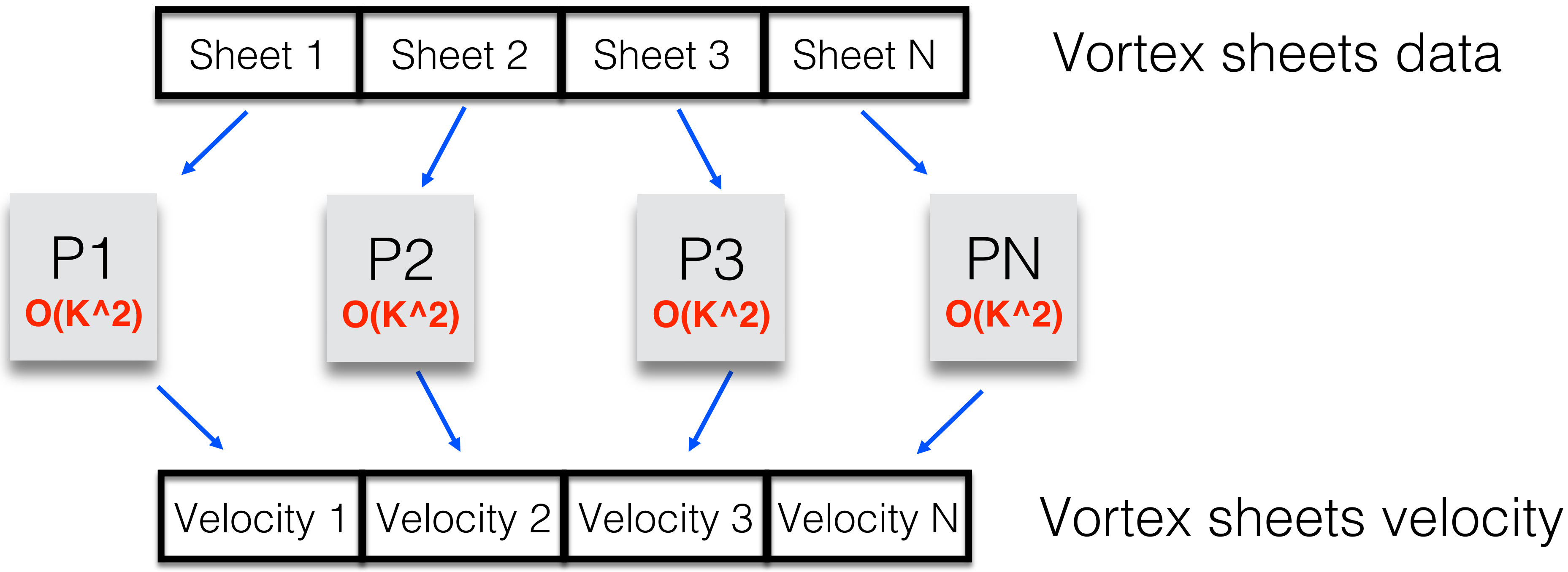


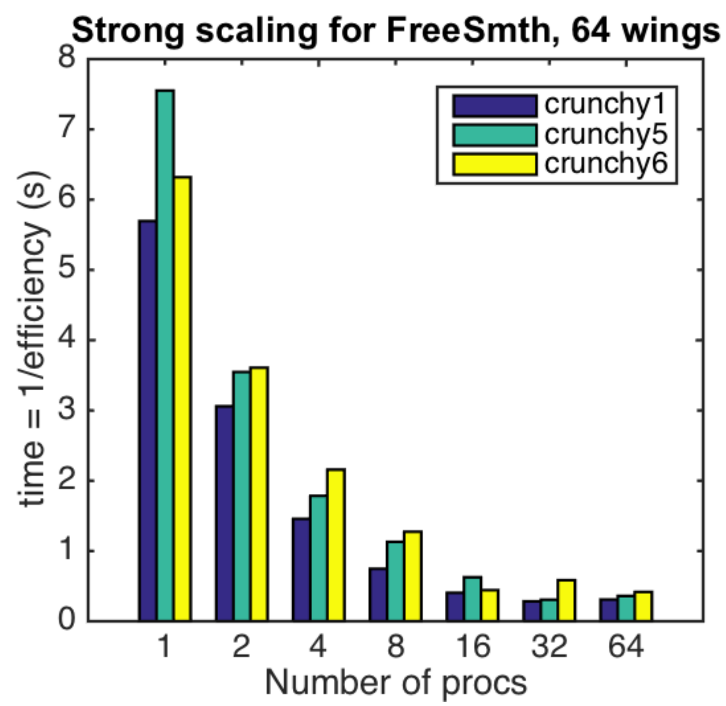
  O(N^2KM) (serial) ⟶ **O(MK) (parallel)** **(in progress…)**

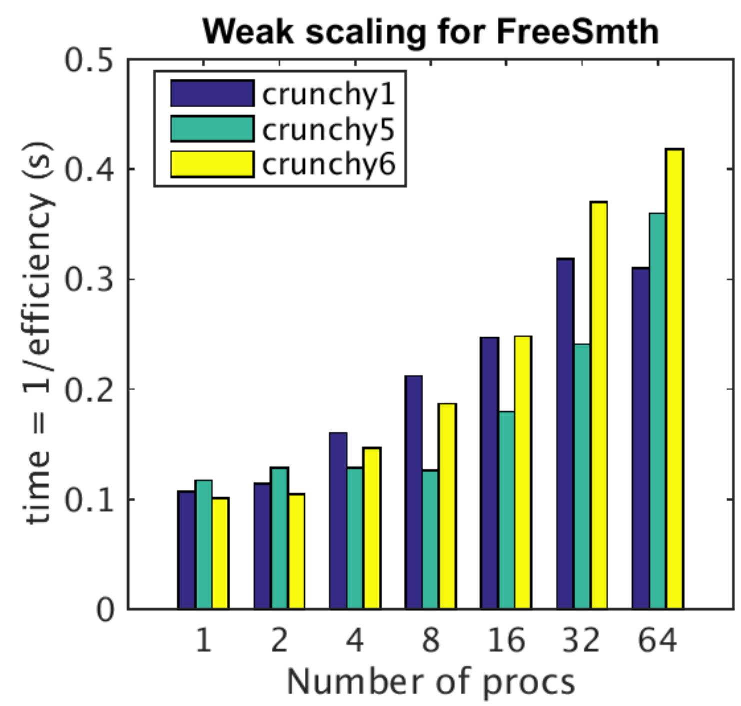
O(NK*(NM+NK)) **serial direct** ⟶ **O(NK*logNK)+O(K^2)+O(MK) ~ O(K^2)** **with FMM and parallelism**

Parallelize vortex sheets self-interaction corrections

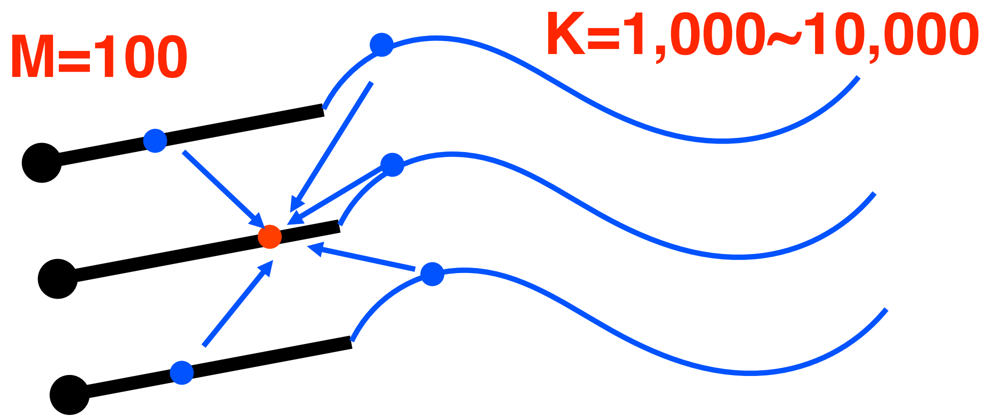
- Generate **N** fake vortex sheets in Matlab, implement OpenMP with C++, compiled into Matlab mex functions. **K=1,000** points on each sheet,







Parallelism of the *implicit* step



# of fish: **N=10~1,000**
Total body points: **N*M=1,000~1e5**
Total vortex sheet points: **N*K=1e4~1e7**

Parallelizing evaluations of velocity on fish bodies

- **N** independent implicit solvers, each solver performs **P** newton iterations
- **NP** calls of single body velocity evaluation, in each call **M** targets, **(N-1)*M+N*K** sources

- Parallel over **N** implicit solvers **(in progress…)**
- Body velocity induced by other bodies — FMM of kernel 1/z, **M** targets, **(N-1)*M** targets

O(NM^2) **(direct)** ⟶ less than **O(NM*logNM)** **(FMM)** ✔

- Body velocity induced by vortex sheets — more accurate quadrature: analytically integrate piecewise linear distribution approximation, parallelize **O(MK)** interaction over **N** sheets

O(NMK) **(serial)** ⟶ **O(MK)** **(parallel)** ✔ **(tests shown later)**

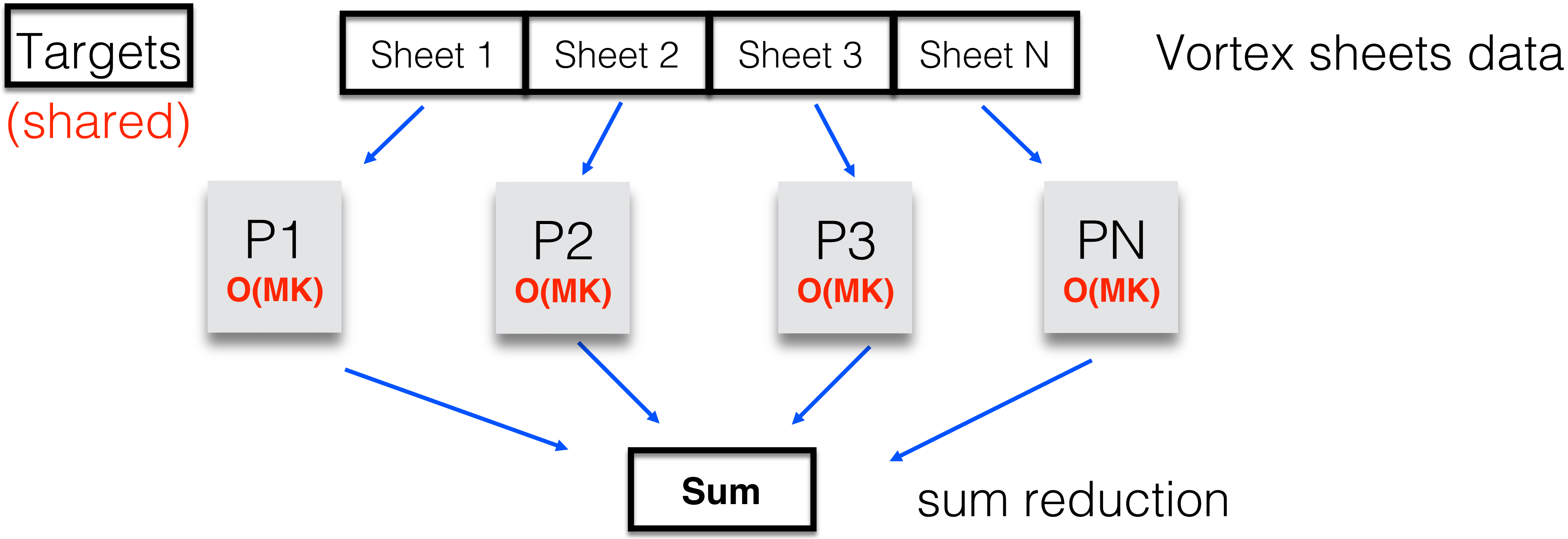
O(NPM*(NM+NK))
serial direct

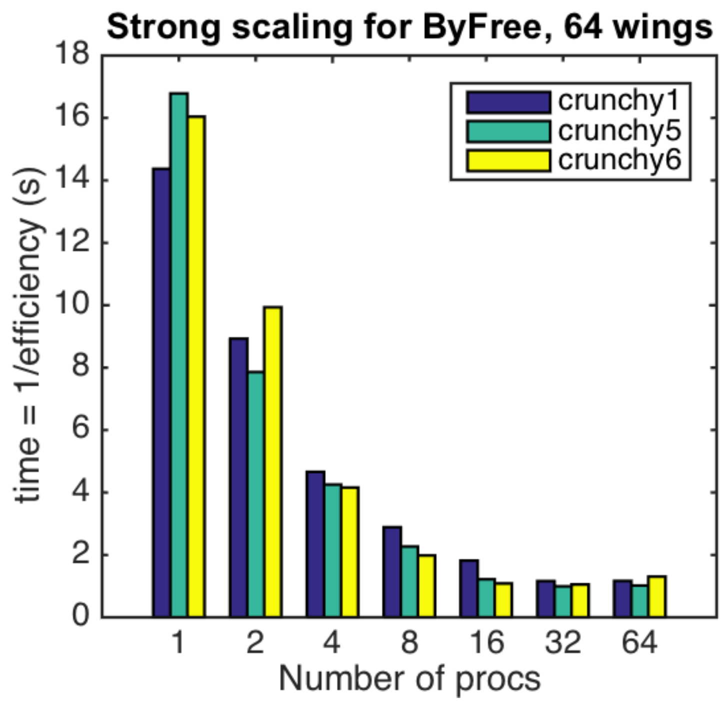
⟶

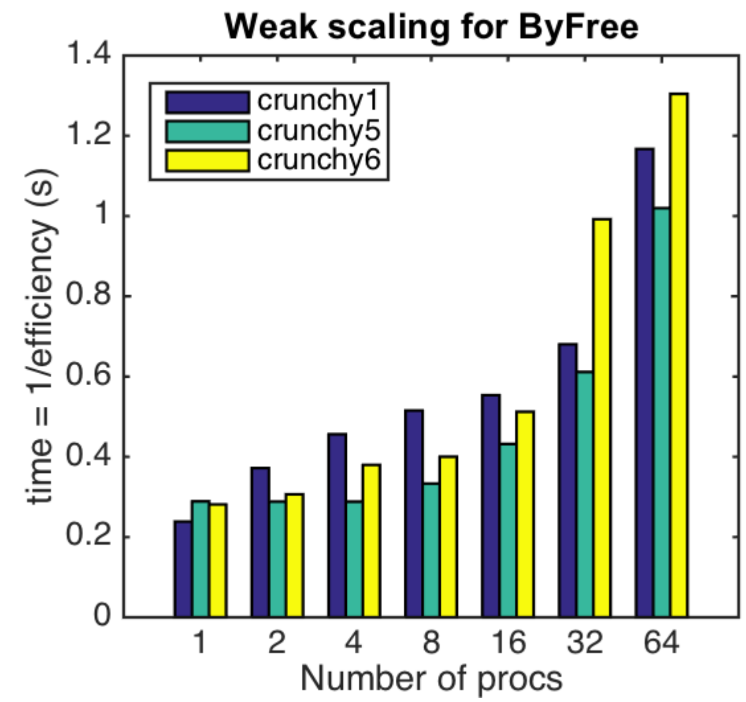
O(PNM*logNM)+O(PMK) ~ O(PMK)
with FMM and parallelism

Parallelize body velocity calculation induced by sheets

- Generate **N** fake vortex sheets and **M=900** target points in Matlab, implement OpenMP with C++, compiled into Matlab mex functions. **K=1,000** points on each sheet.







Outlook

- Matlab code with current parallelisms is about 5 times faster, for N=10 fish.
- Complete the parallelisms, implement for larger fish school
- Use kernel independent FMM, or quadrature by expansion (QBX)

What have I learned?

- Compile C++ OpenMP code into Matlab *mex* function

```
mex -I"../Fish/" -I"../eigen/" -v -largeArrayDims FreeByFreeSmth_mul.
cpp FreeByFreeSmth_mul_routine.cpp -lmwblas -lrt CFLAGS="\$CFLAGS -
fopenmp" LDFLAGS="\$LDFLAGS -fopenmp" CXXOPTIMFLAGS="\$CXXOPTIMFLAGS -
fopenmp"
```

- OpenMP sum reduction with overloaded operator

```
#pragma omp parallel
{
    VectorXd private_r = VectorXd::Zero(nt,1), private_i = VectorXd::Zero(nt,1);

#pragma omp for nowait
    for (int ib = 0; ib < Nw; ++ib){
        VectorXd temp_r = VectorXd::Zero(nt,1), temp_i = VectorXd::Zero(nt,1);
        byfree(zztreal, zztimag, W[ib].zetaf_r, W[ib].zetaf_i, W[ib].gammaf, temp_r, temp_i);
        private_r += temp_r;
        private_i += temp_i;
    }

#pragma omp critical
    {
        pot_r += private_r;
        pot_i += private_i;
    }

}
```

- C++ linear algebra package: Eigen.